Text Retrieval via Semantic Forests

Patrick Schone, Jeffrey L. Townsend, Thomas H. Crystal*, and Calvin Olano
U.S. Department of Defense
Speech Research Branch
Ft. George G. Meade, MD 20755-6000

I. INTRODUCTION

We approached our first participation in TREC with an interest in performing retrieval on the output of automatic speech-to-text (speech recognition) systems and a background in performing topic-labeling on such output. Our primary thrust, therefore, was to participate in the SDR track. In conformance with the rules, we also participated in the *Ad Hoc* text-retrieval task, to create a baseline for comparing our converted topic-labeling system with other approaches to IR and to assess the effect of speech-transcription errors. A second thrust was to explore rapid prototyping of an IR system, given the existing topic-labeling software.

Our IR system makes use of software called Semantic Forests which is based on an algorithm originally developed for labeling topics in text and transcribed speech (Schone & Nelson, ICASSP '96). Topic-labelling is not an IR task, so Semantic Forests was adapted for use in TREC over an eight-week period for the *Ad Hoc* task, with an additional two weeks for SDR. In what follows, we describe our system as well as experiments, timings, results, and future directions with these techniques.

II. GENERAL SYSTEM OVERVIEW

In order to do database designing and then querying, our overall system required a number of steps, as illustrated in Fig. 1. The preliminary steps, shown as the first three blocks in the figure, involve preparing the data for use with our topic-labelling software, Semantic Forests. Since Semantic Forests has not been specifically tailored for SGML applications, each database document needed first to be filtered to select out only the <TEXT> portion of each database message and then formatted for processing. Subsequently, all the words in a file which are spellings of numbers were converted to their numeric value (such as "seven" becoming "7"). For the *Ad Hoc* task, most numbers were already converted, so this stage was eliminated, although it was used for the SDR task. The second preparatory stage was to transform multiword units into single tokens (e.g., "United States" becomes "united_states"). The list of multiwords used for this process was derived both by hand, as well as from frequencies and document frequencies of commonly occurring tuples. The output from this word-joining software became the input into the main engine of Semantic Forests.

*T.H.Crystal is an integree from the IDA Center for Communications Research, Princeton, NJ.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **NOV 1997**	2. REPORT TYPE	3. DATES COVERED **19-11-1997 to 21-11-1997**
4. TITLE AND SUBTITLE **Text Retrieval via Semantic Forests**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Department of Defense,Speech Research Branch,Ft. George G. Meade,MD,20755-6000**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES **Sixth Text Retrieval Conference (TREC-6), Gaithersburg, MD, November 19-21, 1997**		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**13**	

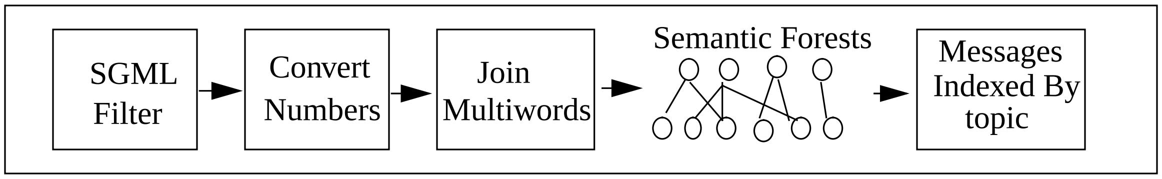
Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Semantic Forests is conceptually simple. Greater detail about the algorithm will be given later. Yet suffice it for now to say that Semantic Forests reads in every word of a document, finds that word (or its stem) in an electronic dictionary, builds a topically-weighted tree based on the definition and frequencies of the word, and lastly, merges all trees together into a common graph, enhancing the scores of the words common across multiple trees. Its output is the sorted *N* highest scoring words from the common graph along with their weights. These *N* words can either come directly from text or may be generalizations or related words from the dictionary. Our goal in using Semantic Forests was to process each database message and derive its topic list, where the list itself, rather than all the words in the message, would be used to index the data.

The final stage for database-building is for an index to be built on these topic lists. Our method for doing the indexing is perhaps fairly standard. Essentially, for each unique word encountered in any topic list, there is a linked list associated. The linked list contains the message numbers, topical scores, ranks, and indicator flags for each message having that word in its topical listing. For clarity sake, it may be useful to mention that a word *X* has a different score and rank for each message in which it appears. To reduce the amount of seek time, each node of the linked list actually stores the information for up to four messages. This, then, completes the steps of database construction.

The methods for building both our *Ad Hoc* and SDR submissions followed this same structure, as shown in Fig. 1. Description of how we used the database to do queries will be discussed in Section V of this paper.

Fig. 1: Overall Processing Methodology



III. SEMANTIC FOREST DETAILS

Having given a general overview of our system, it may be valuable to give a more detailed view of Semantic Forests. As mentioned before, each word in a document is identified in a large recursively-closed electronic dictionary. Each of these input words may be thought of as the root node of a tree. This root node is assigned a value based upon the word's message frequency (f_{word}), a part of speech score (β_{word}), its frequency in a large corpus (F_{word}), a confidence measure of its correctness (K_{word}) and a frequency at the 50%-area mark (F_{max}) of the large corpus' *cumulative word frequency distribution*. Supposing the frequency of words were sorted from lowest to highest, the cumulative word frequency of the *i*th word in the sorted list is the sum of the frequencies of all words of indices less than or equal to *i*. If a salience score for each word is given by the near-smooth function,

$$S(word) = \begin{cases} \varepsilon + \frac{F_{word}}{T}\left(\log\left(\frac{F_{max}}{T}\right) - \varepsilon\right) & F_{word} \le T \\ \log(F_{max}/T) & T < F_{word} \le T^2 \\ \log(F_{max}/(F_{word} - T^2 + T)) & T^2 < F_{word} \le F_{max} \\ 0 & F_{max} < F_{word} \end{cases}, \quad (1)$$

where ε and T are arbitrary values, representing the desired salience of a zero-frequency word and a reliability threshold frequency, then the overall score used for weighting each input word is

$$score(rootWord) = K_{word}(\beta_{word}S_{word})^{f_{word}} . \quad (2)$$

For the *Ad Hoc* task, K_{word} was set to one for every word. On the other hand, for the SDR task, the word error rate (WER) was estimated at 30%, so K_{word} was estimated by the formula $K_{word} = 1 - (0.3)^{f_{word}}$.

If an input word could not be located in the dictionary, however, rather than using ε as its salience, we pretended the word had a training frequency T. If the word was capitalized, it was assumed to be a proper noun, and otherwise it was assumed to have β_{word} of zero unless f_{word} was greater than one (in which case, β_{word} was set to be the score for nouns).

The "large corpus" of preference to be used for identifying training frequencies would have been the whole *Ad Hoc* database set. Due to time constraints, though, we were forced to use and hand-tune only the frequencies from the SDR training corpus in addition to data from some internet discussion groups. It is unknown what adverse effect this may have on system performance, but our speculation is that this caused some slight degradation.

The next step of Semantic Forests is that the words that define each input word are considered to be its children, and they receive as their score a fraction of their parent's value, based again on their individual parts of speech and large-corpus frequency, as well as on a propagation attenuation coefficient (W), their dictionary frequency (d), and the dictionary equivalent of an F_{max}, namely d_{max}. In particular, the fraction of weight that the jth child of the ith word receives is

$$fraction(i, j) = WD_j / \left(\sum_{\forall child(i) = k} D_k \right), \quad (3)$$

where D is a dictionary salience given by

$$D_j = \beta_j (S_j \log(d_{max}/d_j))^{\frac{1}{2}} . \quad (4)$$

This means that the score for the ith child word of parent word j is

$$score(\text{child word } i) = fraction(i,j) * score(\text{parent word } j). \quad (5)$$

This tree could continue to grow by augmenting the children of children and their corresponding scores, etc., but this augmentation was not done for either TREC task. (For additional details, see [1].) As can be seen, though, each word of the message is given a corresponding semantic tree structure, where the input word is the root of that tree.

The last topical step, then, is to merge all of these semantic trees into a common graph and give each word *j* a score *OS(j)* as a function of its graph connections. This is illustrated in Fig. 2. As this is done, words that occur in multiple trees are strongly enhanced and are likely to be topic words or words related to the topic. Let the topical score of word *j* for tree *i* be denoted by *TS*(*i*,*j*). Since word *j* may occur multiple times in a tree, we add the score for each occurrence to get *TS*(*i*,*j*). Let *SUM*(*j*,*q*) be the sum over *i* of $(TS(i,j))^q$. Then the overall score for word *j* is given by:

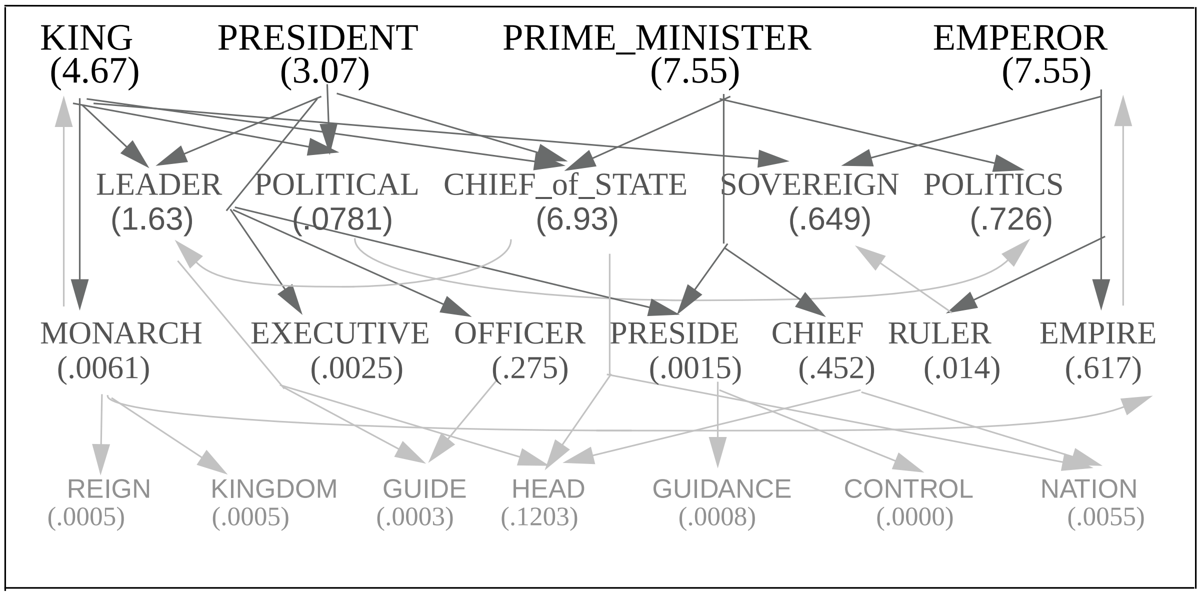
$$OS(j) = D_j^{\frac{1}{2}(\phi(j,1)+\phi(j,2))-1} \cdot SUM(j,1), \tag{6}$$

where the ϕ values come from complicated, *ad hoc* functions based on *n*th-order moments and maximal *TS* values for *j*. Letting *n*(*j*) represent the number of trees where the TS value for *j* is non-zero, and letting *max*(*j*) be the largest of these values, then ϕ can be given as

$$\phi(j,p) = n(j)\left(\frac{1+(-1)^p}{2} - \frac{1}{max(j)} \sum_{q=1}^{p} (-1)^q \frac{SUM(j,q)}{SUM(j,q-1)}\right), \tag{7}$$

where *p* is the maximum number of moments used

Fig. 2: Final weighted graph derived from merging a particular Semantic Forest.



After the common graph is created, the elements of the tree with the largest values can be selected as topics or words that are topically related. However, some pruning of the tree may be desirable.

The type of pruning we used was this: if a word of the graph is not an input word and if $n(j)$ is one, then the word is pruned from the tree. Other types of prunings are possible, but this is the only one performed for TREC purposes. After pruning, the N largest values are selected as topical representatives for the particular document, and are stored with their scores and their rank amongst the N. As is evident, this implies that words may be used in the topic descriptor even though they were not in the original document, so an indicator flag is also stored to indicate if the word was an input word or not.

A comment should be made here about implementation and memory requirements. For the sake of speed as well as memory, the results of forming a common graph were implemented as arrays of words as opposed to large sets of trees. Thus, some of the nice features about trees and graphs are lost, but most of the topical scoring ability is still retained. An example of the contents of a final array (in sorted order) is illustrated in Fig 3. For processing a single document (or query), the equivalent of eight floating point arrays the size of the dictionary (about 38K words for this task) need to be retained in addition to the dictionary itself, as well as information about any new words. This means there is a fairly low RAM overhead (about 4-7 Mbytes). On the other hand, considerable disk space is required to store the Semantic Forests output in uncompressed text form for the whole collection. For the Ad Hoc task, the required storage space was 2.38 Gbytes and for the SDR task it was 4 Mbytes. The indexing on the topical outputs from each message took much less storage. For the *Ad Hoc* task, the indices required 473 Mbytes to store; but for the SDR task, the size was only 3 Mbytes.

Fig.3 Final, sorted topic array

EMPEROR	7.547
PRIME_MINISTER	7.547
CHIEF_of_STATE	6.926
KING	4.675
PRESIDENT	3.071
LEADER	1.634
POLITICS	0.726
SOVEREIGN	0.649
EMPIRE	0.617
CHIEF	0.452
OFFICER	0.275
PRINCIPAL	0.147
ENFORCEMENT	0.142
HEAD	0.120
POLITICAL	0.078
AFFAIR	0.062

IV. TIMINGS on TRAINING

With the system structure built, the next objective was to begin processing the database messages. This was a *very* time-consuming procedure. The filters and pre-processing stages of this task were implemented in Perl scripts, making them easy to write but incredibly slow. Semantic Forests and the indexer, on the other hand, were C code optimized with a -O2 option on SunOS5. Unfortunately, since our processing on the *Ad Hoc* data was done in the week of submission, we inadvertently failed to capture the actual CPU timings for the processing. However, we do have approximate serial wall-clock timings which we believe are similar or on the high side of the actual processing time. For the pre-processing and topic-labelling stages of the *Ad Hoc* task, Table 1 on the next page gives these timings for the full 1997 *Ad Hoc* database material. The wall-clock time for indexing the topic lists was a much faster process, taking 209 minutes. Based on this fact and on the entries in Table 1, the total time for processing was approximately **85.6** hours. For the SDR task, we had two extra weeks to prepare, so we did retain actual system timings for processing. Table 2 gives the timings required for processing the smaller SDR task.

Of course, of interest is the actual platform(s) being used for processing. The two machines used for the *Ad Hoc* processing were a Sun Enterprise 5000 (250 MHz, 8 heads, and 1040 MB of RAM), which is listed in the table as machine type 1, and a SunSparc 20 (100 MHz, 2 heads, and 512 MB of RAM). Only machine type 1 was required for the smaller SDR task.

Table 1: Approximate Wall-clock Time for Topic Labelling on Ad Hoc Task

DOC TYPE	# messages	Machine Type	Pre-proc &Topic Labelling
CR-E	11,358	1	141 min.
CR-H1	7,425	1	96 min
CR-H2	9,139	1	112 min
FBIS1	61,578	1	417 min
FBIS2	26,438	1	174 min
FBIS3	42,455	2	735 min
FR1	26,843	1	201 min
FR2	28,787	1	302 min
FT1	76,857	1	683 min
FT2	133,301	1	897 min
LA1	43,803	1	361 min
LA2	54,603	1	495 min
LA3	34,210	1	311 min

Table 2: System Timing for Full Training on SDR Task

TASK	Machine Type	Pre-proc & Topic Labelling	Indexing
LTT data	1	447.89 sec	11.97 sec
SRT data	1	467.55 sec	14.20 sec

V. PERFORMING QUERIES

The next issue is to address the querying process itself as well as the scoring metrics that correspond to querying. All of the query processing was done automatically with no human intervention. For a given query, a query information extractor was first applied to the raw input which does the equivalent of what the SGML filter did before, while also using a number of regular expression searches to eliminate common non-topic phrases such as “a required document would have,” as seen in the queries of previous years. Afterward, the number conversion, the multiword processing, and Semantic Forests are applied to each of the queries in the file. Thus, for each query, a topic listing is created, almost exactly as had been done for the database messages.

The goal is to determine how correlated the query topics are with each message in the database. Actual implementation of our querying scheme looks up each single token that was output of the query topic list and then finds all messages where that token was in the topic list as well. “Token,” in this instance may either be a single word or a multiword unit. After each token is processed from the query topic list, though, it is as if two vectors have been correlated (namely, the vector of topics from a particular database message and the query topic list). Rather than describing what happens with each token, we will simply show how the two vectors might correlate or agree. Our score is given by

$$agreement = (hits)^{p1}\left(\sum_{\forall hit} mw^{p2} \cdot idf^{p3} \cdot F(mt, qt, mr, qr)\right). \tag{8}$$

The variable *hit* indicates a topic token agreement between the database message topic list and that of the query, and *hit*s, then is just the number of these agreements over *M*-long topic lists (*M* is less than or equal to *N*). An agreement between a multiword unit should in general be worth more than single-word agreements, so the value *mw* is introduced. The value *mw* is set to 1 if the token in agreement is a single word, and otherwise it is the number of words in a multiword unit. Likewise, if the token in agreement is rare in terms of the number of topic lists it appears in, it should have more weight than a more frequent word. Thus, *idf* is introduced, which is simply the inverse document frequency of a word as applied to the topic lists. The function *F* is user-specified, as are the parameters p_i. *F* takes into consideration the topic ranks and the topic scores from the agreement, and is the key component of the agreement score. The variables *mt* and *qt*, represent the topic score of the particular token in the data message and in the query, respectively; likewise, *mr* and *qr* represent message and query ranks.

The values *mt* and *qt*, when output from the topic algorithm, are L_2-normalized. The rationale for this normalization was that we might want to later take a product between two topic lists and make the inner product of a list with itself be equal to one. With this notion in mind, then, the experimentation we did to develop the *F* function basically limited its structure to be

$$F(mt, qt, mr, qr) = warp((mt \cdot qt)^{p4} \otimes H(mr, qr)), \qquad (9)$$

where $\otimes$ indicates some simple commutative function such as sum or multiply, "warp" indicates some non-linear function, and *H* is a type of correlation function. It seemed useful to turn rank scores into alternative types of topic scores. This can be achieved by normalizing the reverse ranks (counting backwards from *M)*; that is, if a rank is y, the normalized reverse rank would be

$$y' = (M+1-y)/BT. \qquad (10)$$

The normalizing value *BT* is simply the square root of the sum of squares over the integers ranging from 1 to *M.* Thus, if we force the ranks to behave similar to the actual topic scores, the function *H* can be rewritten

$$H(mr, qr) = (mr' \cdot qr')^{p5}. \qquad (11)$$

The actual final formulas and other experimentation that were performed will be explained in the next section. Table 3 shows the time it took to perform retrieval for both the *Ad Hoc* and the SDR tasks. It includes forming the topic lists for the queries and performing the score calculations. For the conditions under which we ran, wall-clock time is assumed to have been only slightly greater than CPU time.

Table 3: Query Time for Topics 301-350 and Topics SDR1-SDR50

TASK	Machine Type	Time	Timing Method
Ad Hoc: Description Only	1	11 min	wall clock
Ad Hoc: All topic info	1	26 min	wall clock
LTT: All topic info	1	8.41 sec	CPU clock
SRT: All topic info	1	8.66 sec	CPU clock

VI. EXPERIMENTATION

For our system, one of the primary 'experiments' was to get it to give reasonable answers. An initial system prototype applied to the SDR task suggested that our methodology had some potential, so construction of the full indexing-querying software was begun. We experimented with our system by using the query set from 1996 and querying against only those parts of this year's data that were also available in 1996l. All of our runs were completely automatic, so we were able to make many experiments. We compared our results against the TREC5 short and long automatic columns listed in Table 1 of Karen Sparck Jones' "Summary Performance Comparisons TREC-2, TREC-3, TREC-4, and TREC-5," noting that 25% was the lowest average precision listed for retrieval of 30 documents. As we practiced this *Ad Hoc* task, we were able to move our average precision on the top 30 documents retrieved on long automatic queries from a mere 4% at the onset to 19% at the end of experimentation. Likewise, in the SDR task, using the six sample queries, our initial protoype had an average rank of 45 and inverse of the average inverse rank of 1.5, both of which we were able to bring down substantially, getting scores as low as 4.7 and 1.37 respectively. This was done using a set of queries which included the six supplied by NIST and an additional 40 of our own queries. Thus, in this experimental phase, we learned many useful things that will also be commented upon here.

INITIAL COMMENTS

Three comments are in order at the onset. The original application of Semantic Forests, as noted previously, was to automatically label the topic of a document. This, though, is not the case for queries. When a person makes a query, he or she may not necessarily want to retrieve something whose topic matches the query. Having looked at the messages marked by NIST as relevant against some of last year's queries, we noted that some documents were so marked even if they only made passing reference to the query word or phrase. These instances would be clear losses to us since the fact of a word being in a document would not necessarily guarantee that the word would also be topical.

The second comment is that as we proceeded to do our experiments for the *Ad Hoc* experiments, we recognized that our software had two programming bugs. When these bugs were corrected, we observed a boost in our overall average precision on the top 30 (APRT30) documents by an absolute 7.5%.

Thirdly, since we were only using a subset of the total 1996 data as we performed our experiments, we expected we would be losing a few percentage points due to the fact that some of last year's queries only had documents that appeared in early databases, meaning we would automatically lose in those situations. Our expectation was that whatever our final APRT30, we could probably improve performance by 2-3% based on the fact that we would have the full 1997 data set. Also, since our output would actually be evaluated, we thought this might contribute an additional 1-2%.

EXPERIMENTS PERFORMED

All the experimentation that was performed for the *Ad Hoc* task involved either specifying the functions and parameters mentioned in section IV, or limiting or expanding the number of non-input synonym words that are generated by Semantic Forest for either the database document or the query. Our SDR experimentation also involved trying to find common errors and omissions made in the recognizer output and supplementing our electronic dictionary to take these variations into account.

Synonymy:

A common recurring theme in past TREC's is how to supplement a query with synonyms in such a way that there is a performance boost. The idea of producing conceptually similar words is something quite natural for Semantic Forests. We expected that this innate quality would be a big boon for our system. Unfortunately, Semantic Forests reports terms as found in its dictionary which, though conceptually similar, are not always synonymous. Likewise, Semantic Forest does not yet report which sense of the word is desirable. Both of these factors made it difficult to use synonyms. The general finding when using the out-of-the-box synonyms were that the messages that were already fairly well correlated suddenly had gigantic scores; but weakly correlated messages were not enhanced and were generally retrieved lower in the queue since other documents might have more synonyms. An example of both of these cases might be made from the SDR sample 6 queries. Before synonyms were added, our best query had an agreement score of 98 and a rank of 1, while our worst had a score of 0.58 and a rank of 63. After the synonyms were added, the first now had a score of 500K and the rank did not change; but the worst had a slight drop in score of 0.367 (due to the fact that synonyms can possibly rank higher than input words), and, more importantly, its rank had dropped to 127. Thus, for this TREC, we limited our synonyms to two other types.

The first was to use subcomponents of multiword units. In this instance, if a query has a multiword unit that agrees with one of the messages, there is a hit not only on the word itself but on the salient subcomponents as well. On the other hand, a database message may not have the particular multiword being sought but may have the component words. For the *Ad Hoc* query, we did not use this, though it was incorporated into the SDR experiments and resulted in dropping the inverse of the average inverse rank on our 45-query set from 1.45 to 1.37 and average rank from 5.83 to 5.58 at the particular time it was first added.

The second type of synonym has to do with adjectival nouns that reference a country. For example, "French" implies "France" and "Israeli" implies "Israel." Therefore, if an adjectival noun existed in a document and if it had in its definition a country name with the first two characters the same, the country name was added to the document and processed as if it had actually been a member of the text. For the *Ad Hoc* experiment, we used this type of synonymy on both the database documents as well as on the queries, and we experienced a positive effect. On the version of the system used to conduct the experiment, the APRT30 increased (on the full topic) from 17.5% to 19.0%. On the SDR task, this same synonymy was applied to both the query and document and to the query alone. There was a decrease in average rank when the synonymy was limited to only the query.

Parameter Modifications

For both the *Ad Hoc* and the SDR tasks, determining the best user-specified parameters and functions was difficult, but fairly useful. For the *Ad Hoc* task, the parameter modifications did results in slight improvements. We basically let $p2=0$, $p1=p3=p4=p5=1$, warp=$\sqrt{x}$, and $\otimes$ = multiply in equations (8), (9) and (11). We started with M=250 and tried to reduce this, but this change decreased the precision. The biggest improvements for the *Ad Hoc* came in limiting the allowable ranks. We got a 0.8% absolute improvement in APRT30 when we set qr' to zero when $qr>75$ (i.e., weeding out lesser important query topics). Also, when a hit occurred such that $mr>6$, it was counted as only 1/4 in the *hits* parameter. This gave an absolute 1.2%.

However, for the SDR task, these parameters made much larger differences. In particular and of prime interest is the fact that we eventually ended up deciding that the topic score was of no importance when compared with the rank, and it was completely eliminated. For the final system, we settled on letting $p1=3$, $p2=2$, $p3=1.5$, $p4=0$, and $p5=1$; $\otimes$ remained a simple multiply, but warp(x) was changed to $(0.85)^x$ and H, as previously defined in Equation (11), was modified to become

$$H(mr, qr) = (mr + qr)^{p5}. \qquad (12)$$

Common Recognizer Errors

As was mentioned before, we were particularly interested in the SDR track of TREC97. Our hope was that Semantic Forests could potentially knit together the true topics of errorful transcriptions, but we also hoped to be able to supplement that effort by locating commonly misrecognized words and putting into the dictionary the misrecognition. In particular, we wanted to supplement the dictionary with words that are high frequency in the LTT training files, but non-existent in the SRT. After analyzing the training data, we realized that there were not many instances of this kind of phenomenon, but the words that were missing from recognition were usually critical. In particular, “Netanyahu,” “Valujet”, “Freemen,” and “Admiral Boorda” are very topical words that appeared in many instances across the LTT files, but never occurred in the SRT. Common misrecognitions of “Valujet,” for example, were the phrases “value jet” and “valued jet;” “Netanyahu” often was recognized with the word “neon” in it, such as “neon who;” “Freeman” almost always appeared as two separate words; and “Boorda” often appeared as “border” or something similar. As an experiment, we wrote some practice queries that only involved these phrases and compared our algorithm’s output to that of another retrieval system. As one might expect, our performance was far better. Unfortunately, after we had submitted our actual evaluation of SDR, we found that none of these words were in the SDR queries. On the other hand, the word “Unabomber” did appear in the queries and we were prepared.

VII. OTHER INTERESTING OBSERVATIONS

It is interesting and useful to make note of a few other observations that can be made about this approach. These areas are things that are inherent in the algorithm but may be considered experimental in their own right. These are the areas of stemming, use of numbers, and boolean logic.

Stemming

In the past, different groups have experimented with different kinds of stemming, such as using the first *k* characters or using some more sophisticated method. For Semantic Forests, stemming is automatic. If a word in the text document is also in the electronic dictionary, then the word is considered to already be stemmed. There are instances when this consideration is wrong, where, for example, what might be a conjugation of a verb might also have another meaning, and therefore the conjugation may also be stored in the dictionary. Yet these cases are infrequent. On the other hand, if a word is not found in the dictionary, then procedures are applied to see if it is a different word form of one of the words already in the dictionary. If the word is still not found, it is assumed to be a new word.

However, words that are considered to be new result in a great difficulty that does not exist in other stemmers. Suppose, for example, that a words “cryogenically” and “cryogenics” appear in training data but are not words that existed in Semantic Forests’ dictionary (nor their stems). If a query is made about “cryogenics,” only messages with that exact word will be identified. We did not realize the full extent of this problem until the SDR evaluation, where for some reason, the word “programmers” was not known to Semantic Forests. It therefore stored the whole word and missed any related or partial words, causing our recognition of that message to be abysmal.

Numbers

As was mentioned before, a number conversion routine is one of the early precursors to Semantic Forests. Semantic Forests knows what numbers are, and in fact, it can interpret years to a certain extent, even knowing some major events of those years. In the 1996 evaluation, one of the queries had asked to find some event “since 1950.” Semantic Forests knows both words, but it does not understand the pairwise construct. It has not yet learned to interpret “since <date>” as “greater than or equal <date>.” We did not have time to put this into Semantic Forests, so we made an attempt to fake it in the SGML filter. The phrase “since <date>” was converted to the string “<date>,..., 1996, 1997,” i.e., “since 1950” became “1950, 1951,..., 1997.” Semantic Forests reported that dates and numbers were the key topics. This clearly was an incorrect interpretation. As a result, for the final system, we decided to leave out any special numerical processing other than what was already in the system.

Boolean Logic

Similarly, boolean logic is something that is not yet interpreted by Semantic Forests, so the “not” logic adversely effects performance of the retrieval system. To partially remedy this, the SGML filter was told that if it saw “word1 not word2,” that it should just eliminate word2 altogether. Other rudimentary facilities were added to this script which enhanced our whole query routine’s ability to find documents...at least in training. Yet the other boolean constructs were basically unregarded, which could have caused negative effects on processing.

VIII. FUTURE WORK

There were a number of areas that we would like to explore but did not have the time prior to evaluation to properly pursue. Other ideas did have some initial attempts made, but though the ideas may eventually provide great benefits, these first attempts were unfruitful. In particular, then, the areas we would like to work on in the future would be:

[1] Take full advantage of the synonym property of Semantic Forests. A place where this would be of particular utility is when a query is performed and there is a particular word in the query that has no messages containing it. This would have been a sure win on the SDR task, since this potentially helps reduce the number of catastrophic failures that might arise;
[2] Apply on an *Ad Hoc* task the multiword decomposition, and use the adjectival nouns only on the queries themselves, where both of these resulted in improvements on the SDR task;
[3] Insert number parsing and boolean logic directly into Semantic Forests; and lastly
[4] Perform a second pass search which looks at the actual words of the document after the topic routine is used to reduce the search set.

IX. REFERENCES

Allan, J., Callan, J., Croft, B., Ballesteros, L., Broglio, J., Xu, J., Shu, H., “Inquery at TREC-5,” Center for Intelligent Information Retrieval, Dept. of Computer Science, University of Massachusetts, Amherst, Mass.

Jones, Karen Sparck, “Summary Performance Comparisons TREC-2, TREC-3, TREC-4, TREC-5”, Computer Laboratory, University of Cambridge, 10 Feb 1997.

Schone, P., Nelson, D., “A Dictionary-Based Method for Determining Topics in Text and Transcribed speech,” 1996 IEEE International Conference on Acoustics, Speech, & Signals Processing, Atlanta, Georgia, May, 1996; Vol. 1, pp. 295-298.